In the Claims

This listing of claims will replace all prior versions and listings of claims in the application:

1. (Canceled)

2. (Currently Amended) ~~The~~ A software development method ~~of Claim 2~~ for debugging software on a target system having a plurality of processors configured with shared memory in which ~~the~~ a software memory bus performs processing of shared memory access requests using a method for transparently writing to shared memory when debugging a multiple processor system, the method comprising the steps of:

receiving user input to define a hardware configuration of the target system;

creating a software memory map of how each of the plurality of processors may access memory including whether each of said plurality of processors may read from and write to a range of memory or may only read from said range of memory;

loading drivers for each of the plurality of processors;

activating a first debug session associated with a first processor of the plurality of processors;

activating at least a second debug session associated with a second processor of the plurality of processors wherein each debug session is operable to transmit read requests and write requests to its associated processor;

processing shared memory access requests via a software memory bus;

detecting a write request to a shared memory location by the first debug session; and

if the first processor associated with the first debug session has write access to the shared memory location

then

selecting the first processor to perform the write request;

else performing the following steps ~~a-b~~ a-c:

a. searching the software memory map to determine if the second processor has write access to the shared memory location;

b. selecting the second processor to perform the write request; and

c. passing the write request initiated by the first debug session to the selected processor for execution.

3. (Original) The method of Claim 2 wherein the step of passing the write request comprises the steps of:

searching the software memory map for a second plurality of processors;

broadcasting the write request to the second plurality of processors; and

performing cache coherency updates in response to the write request in each of the second plurality of processors.

4. (Original) The method of Claim 3 wherein the step of broadcasting the write request comprises indicating that the write request is intended for maintaining cache coherency as opposed to a normal write request.

5. (Original) The method of Claim 3 wherein the step of performing comprises using cache coherency capabilities, if any, of a processor in response to the write request intended for maintaining cache coherency.

6. (Currently Amended) The method of Claim 3 wherein:

the step of creating comprises denoting in the software memory map all the shared memory locations that contain program instructions upon each initialization of the target system;

the step of passing the write request additionally comprises the step of determining that the shared memory location contains a program instruction; and

the cache is an instruction cache.

7. (Currently Amended) ~~The~~ A software development ~~system of Claim 1~~ method for debugging software on a target system having a plurality of processors configured with shared memory, comprising steps of:

receiving user input to define a hardware configuration of the target system;

creating a software memory map of how each of the plurality of processors may access memory including whether each of said plurality of processors may read from and write to a range of memory or may only read from said range of memory;

loading drivers for each of the plurality of processors;

activating a first debug session associated with a first processor of the plurality of processors;

activating at least a second debug session associated with a second processor of the plurality of processors wherein each debug session is operable to transmit read requests and write requests to its associated processor; and

processing shared memory access requests via a software memory bus, in which ~~the software memory bus performs~~ processing of shared memory access requests via the software memory bus uses ~~using~~ a method for maintaining coherency of software breakpoints in shared memory when debugging a multiple processor system, the method comprising the steps of:

setting a first software breakpoint in a shared memory location in the first debug session such that all debug sessions are notified of the setting of the breakpoint; and clearing the first software breakpoint in the shared memory location in the second debug session such that all debug sessions are notified of the clearing of the breakpoint.

8 to 12. (Canceled)

13. (Currently Amended) ~~The~~ A software development ~~system of Claim 1~~ method for debugging software on a target system having a plurality of processors configured with shared memory, comprising steps of:

receiving user input to define a hardware configuration of the target system;

creating a software memory map of how each of the plurality of processors may access memory including whether each of said plurality of processors may read from and write to a range of memory or may only read from said range of memory;

loading drivers for each of the plurality of processors;

activating a first debug session associated with a first processor of the plurality of processors;

activating at least a second debug session associated with a second processor of the plurality of processors wherein each debug session is operable to transmit read requests and write requests to its associated processor; and

processing shared memory access requests via a software memory bus, in which ~~the software memory bus performs~~ processing of shared memory access requests via the software memory bus uses ~~using~~ a method for transparently maintaining cache coherency when debugging a multiple processor system with common shared memory, the method comprising the steps of:

denoting in the software memory map the shared memory locations whether or not each processor of the plurality of processors has a cache;

detecting a write request to a shared memory location by the first debug session;

passing the write request initiated by the first debug session to the first processor for execution;

searching the software representation of the memory map for a first plurality of processors that have read access to the shared memory location;

broadcasting the write request to the first plurality of processors; and

performing cache coherency updates in response to the write request in each of the first plurality of processors.

14. (Original) The method of Claim 13 wherein the step of broadcasting the write request comprises indicating that the write request is intended for maintaining cache coherency as opposed to a normal write request.

15. (Original) The method of Claim 13 wherein the step of performing comprises using cache coherency capabilities, if any, of a processor in response to the write request intended for maintaining cache coherency.

16. (Currently Amended) a method for transparently maintaining cache coherency when debugging a multiple processor system with common shared instruction memory, the method comprising the steps of:

~~denoting in the software memory map the shared memory locations whether or not each processor of the plurality of processors has a cache;~~

~~detecting a write request to a shared memory location by a first debug session;~~

if the first processor associated with the first debug session has write access to the shared memory location then

selecting the first processor to perform the write request;

else performing the following steps a-b:

a. searching the software memory map for a second processor with write access to the shared memory location;

b. selecting the second processor to perform the write request;

passing the write request initiated by the first debug session to the selected processor for execution;

searching the software memory map for a second plurality of processors that have read access to the shared memory location;

broadcasting the write request to the second plurality of processors; and

performing cache coherency updates in response to the write request in each of the second plurality of processors.

17. (Original) The method of Claim 16 wherein the step of broadcasting the write request comprises indicating that the write request is intended for maintaining cache coherency as opposed to a normal write request.

18 and 19. (Canceled)